Kernel Methods

Kernel embedding of distributions and large scale kernel machines

Hachem Kadri, Liva Ralaivola
prenom.nom@lis-lab.fr

LIS, CNRS, Aix-Marseille Université, France

Outline

Kernel Embedding of Distributions
- Example: the 2-sample test problem
- Kernel Embedding of Distributions

Large Scale Kernel Methods
- Random Features for Kernel Machines
- Fast optimization methods

Outline

Kernel Embedding of Distributions
- Example: the 2-sample test problem
- Kernel Embedding of Distributions

Large Scale Kernel Methods
- Random Features for Kernel Machines
- Fast optimization methods

Two-Sample Problem (see [Gretton et al., 2007, Gretton et al., 2012])

Problem

- Let $S_X = \{x_i\}_{i=1}^{n_x} \sim P$ and $S_Y = \{y_i\}_{i=1}^{n_y} \sim Q$
- Question: given S_X and S_Y, is it possible to determine whether $P = Q$?
- Or, is it possible to provide a statistical test to decide $P = Q$:
  - $\mathbf{H_0} : \mathbf{P} = \mathbf{Q}$ (null hypothesis)
  - $\mathbf{H_A} : \mathbf{P} \neq \mathbf{Q}$ (alternative hypothesis)

Two-Sample Problem (see [Gretton et al., 2007, Gretton et al., 2012])

Problem

- Let $S_X = \{x_i\}_{i=1}^{n_x} \sim P$ and $S_Y = \{y_i\}_{i=1}^{n_y} \sim Q$
- Question: given S_X and S_Y, is it possible to determine whether $P = Q$?
- Or, is it possible to provide a statistical test to decide $P = Q$:
  - $\mathbf{H_0} : \mathbf{P} = \mathbf{Q}$ (null hypothesis)
  - $\mathbf{H_A} : \mathbf{P} \neq \mathbf{Q}$ (alternative hypothesis)

Indirect solution

1. Estimate by $\hat{P}$ and $\hat{Q}$ the distributions P and Q (Parzen windows, mixture of Gaussians...)
2. Compute the distance d between $\hat{P}$ and $\hat{Q}$ (L_2, L_1, KL, ...)
3. If d if beyond some threshold then reject $\mathbf{H_0}$

Cons

- Curse of dimensionality
- Choice of the models to estimate the distributions
- Biased estimation of the distributions
- ...

Two-Sample Problem (see [Gretton et al., 2007, Gretton et al., 2012])

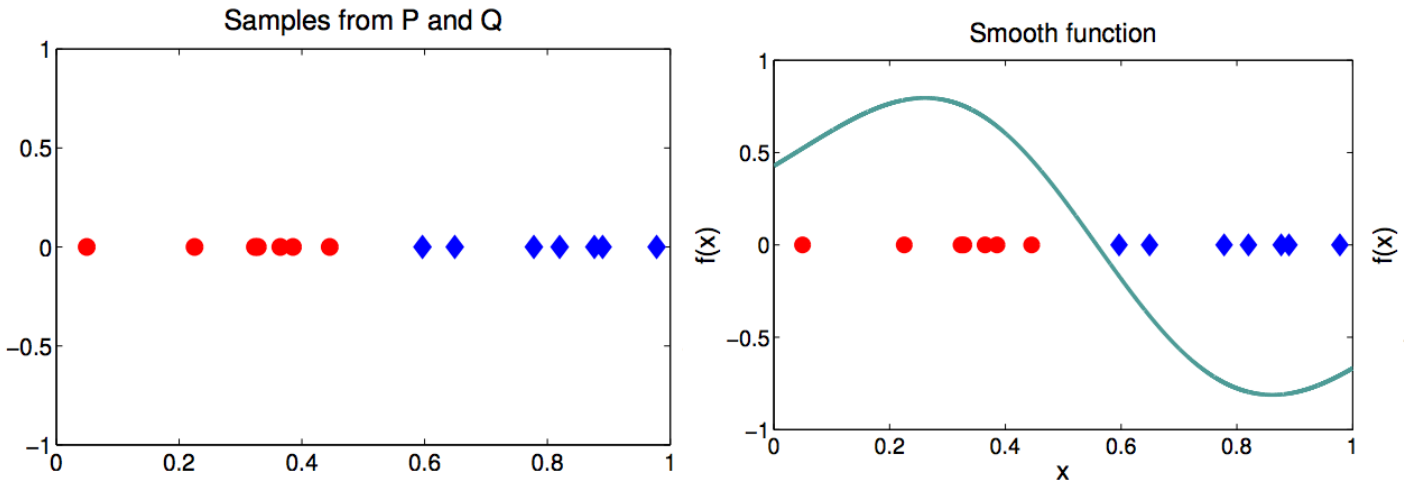
Problem

- Let $S_X = \{x_i\}_{i=1}^{n_x} \sim P$ and $S_Y = \{y_i\}_{i=1}^{n_y} \sim Q$
- Question: given S_X and S_Y, is it possible to determine whether $P = Q$?
- Or, is it possible to provide a statistical test to decide $P = Q$:
  - $\mathbf{H_0}$: $\mathbf{P} = \mathbf{Q}$ (null hypothesis)
  - $\mathbf{H_A}$: $\mathbf{P} \neq \mathbf{Q}$ (alternative hypothesis)

Maximum Mean Discrepancy – MMD [Fortet and Mourier, 1953]

Find a smooth (witness) function that distinguishes P vs. Q

$$MMD(P, Q, \mathcal{F}) := \sup_{f \in \mathcal{F}} [\mathbb{E}_{X \sim P} f(X) - \mathbb{E}_{Y \sim Q} f(Y)]$$



images from D. Sejdinovic, 2014

Two-Sample Problem (see [Gretton et al., 2007, Gretton et al., 2012])

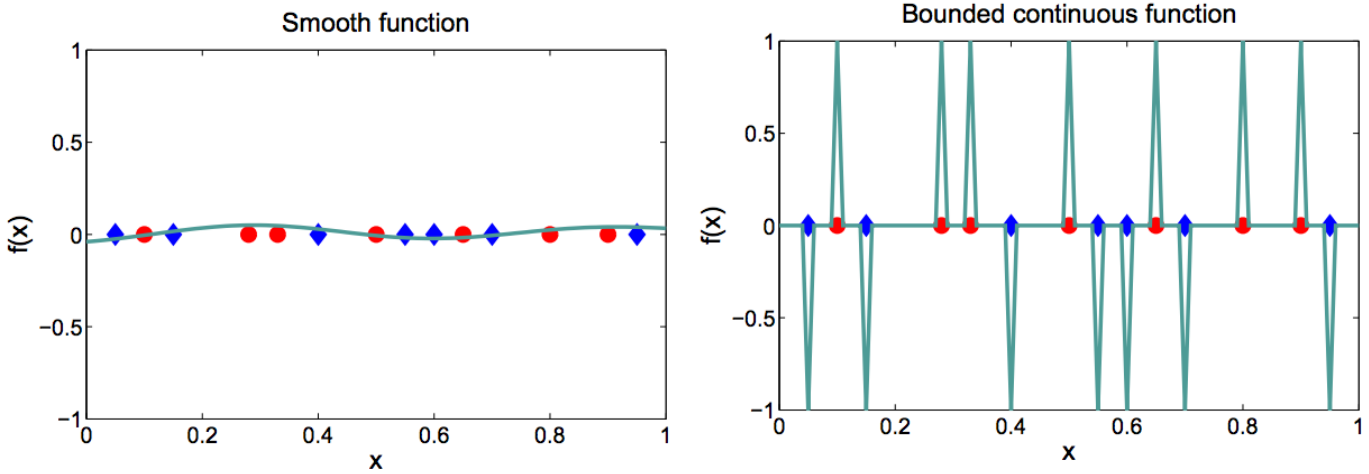
Problem

- Let $S_X = \{x_i\}_{i=1}^{n_x} \sim P$ and $S_Y = \{y_i\}_{i=1}^{n_y} \sim Q$
- Question: given S_X and S_Y, is it possible to determine whether $P = Q$?
- Or, is it possible to provide a statistical test to decide $P = Q$:
  - $\mathbf{H_0}$: $\mathbf{P} = \mathbf{Q}$ (null hypothesis)
  - $\mathbf{H_A}$: $\mathbf{P} \neq \mathbf{Q}$ (alternative hypothesis)

Maximum Mean Discrepancy – MMD [Fortet and Mourier, 1953]

Find a smooth (witness) function that distinguishes P vs. Q

$$MMD(P, Q, \mathcal{F}) := \sup_{f \in \mathcal{F}} [\mathbb{E}_{X \sim P} f(X) - \mathbb{E}_{Y \sim Q} f(Y)]$$



images from D. Sejdinovic, 2014

Theoretical properties of MMD

$$MMD(P, Q, \mathcal{F}) := \sup_{f \in \mathcal{F}} [\mathbb{E}_{X \sim P} f(X) - \mathbb{E}_{Y \sim Q} f(Y)]$$

Theorem

Under mild conditions $\mathcal{H}$ on $\mathcal{F}$:

$$MMD(P, Q, \mathcal{F}) = 0 \Leftrightarrow P = Q$$

Theoretical properties of MMD

$$MMD(P, Q, \mathcal{F}) := \sup_{f \in \mathcal{F}} [\mathbb{E}_{X \sim P} f(X) - \mathbb{E}_{Y \sim Q} f(Y)]$$

Theorem

Under mild conditions $\mathcal{H}$ on $\mathcal{F}$:

$$MMD(P, Q, \mathcal{F}) = 0 \Leftrightarrow P = Q$$

Theorem (Dudley, 84)

$\mathcal{H}$: $\mathcal{F}$ is the space of continous bounded functions on $\mathcal{X}$.

Theoretical properties of MMD

$$MMD(P, Q, \mathcal{F}) := \sup_{f \in \mathcal{F}} [\mathbb{E}_{X \sim P} f(X) - \mathbb{E}_{Y \sim Q} f(Y)]$$

Theorem

Under mild conditions $\mathcal{H}$ on $\mathcal{F}$:

$$MMD(P, Q, \mathcal{F}) = 0 \Leftrightarrow P = Q$$

Theorem (Dudley, 84)

$\mathcal{H}$: $\mathcal{F}$ is the space of continous bounded functions on $\mathcal{X}$.

Theorem (Steinwart 2001, Smola 2006)

$\mathcal{H}$: $\mathcal{F} = \{f : \|f\|_{\mathbb{H}} \leq 1\}$ is a unit ball of a Reproducing Kernel Hilbert Space $\mathbb{H}$, associated with a universal *kernel. (See, e.g. Gaussian kernel, Laplace Kernel, Matern kernel.)*

MMD and kernels

RKHS $\mathbb{H}$, kernel k

- $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ a positive kernel
- $\mathbb{H}$: associated RKHS

Therefore, $\forall f \in \mathbb{H}$

$$f(x) = \langle k(x, \cdot), f \rangle, \ \forall x \in \mathcal{X}$$

(Empirical) Expectations and Kernel Mean Embedding

$$\mathbb{E}_{X \sim P} f(X) = \mathbb{E}_{X \sim P} \langle k(X, \cdot), f \rangle = \langle \mathbb{E}_{X \sim P} k(X, \cdot), f \rangle$$

$$\frac{1}{m} \sum_{i=1}^{m} f(X_i) = \frac{1}{m} \sum_{i=1}^{m} \langle k(X_i, \cdot), f \rangle = \left\langle \frac{1}{m} \sum_{i=1}^{m} k(X_i, \cdot), f \right\rangle$$

- $\mu(P) = \mathbb{E}_{X \sim P} k(X, \cdot) \in \mathbb{H}$ is the mean embedding of P
- $\widehat{\mu}(S_X) = \frac{1}{m} \sum_{i=1}^{m} k(X_i, \cdot) \in \mathbb{H}$ is the empirical mean embedding of S_X

MMD and kernels

Setting $\mathcal{F} = \{f : \|f\|_{\mathbb{H}} \leq 1, f \in \mathbb{H}\}$

$$
\begin{aligned}
MMD(P, Q, \mathcal{F}) &= \sup_{f \in \mathcal{F}} [\mathbb{E}_{X \sim P} f(X) - \mathbb{E}_{Y \sim Q} f(Y)] \\
&= \sup_{f \in \mathcal{F}} [\langle \mathbb{E}_X k(X, \cdot), f \rangle - \langle \mathbb{E}_Y k(Y, \cdot), f \rangle] \\
&= \sup_{f \in \mathcal{F}} [\langle \mu(P), f \rangle - \langle \mu(Q), f \rangle] \\
&= \sup_{f \in \mathcal{F}} \langle \mu(P) - \mu(Q), f \rangle = \|\mu(P) - \mu(Q)\|_{\mathbb{H}}
\end{aligned}
$$

MMD: distance between probabilities in the kernel embedding space

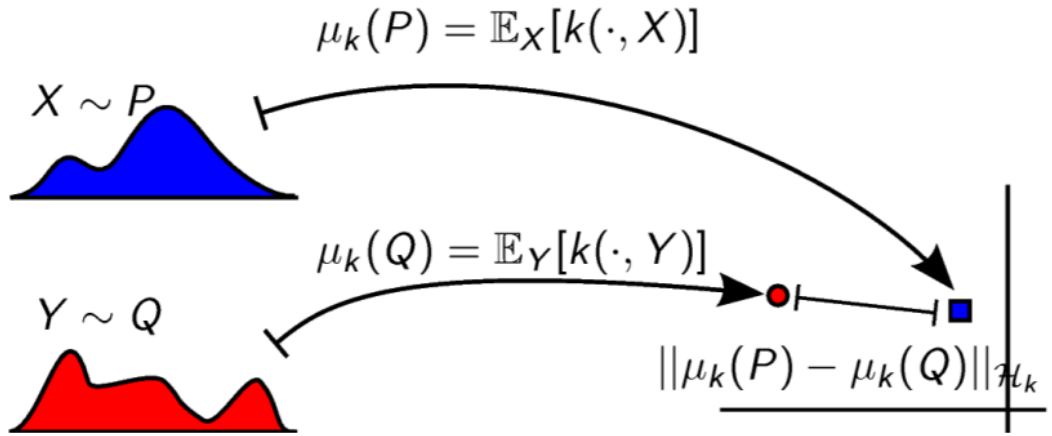


image from D. Sejdinovic, 2014

MMD and kernels

In practice...

- $MMD^2(P, Q, \mathcal{F}) = \mathbb{E}_{X,X'} k(X, X') + \mathbb{E}_{Y,Y'} k(Y, Y') - 2\mathbb{E}_{X,Y}(X, Y)$
- $\widehat{MMD}^2(S_X, S_Y, \mathcal{F})$ defined as

$$\frac{1}{n_x(n_x-1)} \sum_{i \neq j} k(x_i, x_j) + \frac{1}{n_y(n_y-1)} \sum_{i \neq j} k(y_i, y_j) - \frac{2}{n_x n_y} \sum_{i \neq j} k(x_i, y_j)$$

  is an unbiased estimator of MMD^2.
- If $n_x = n_y$, then $\widehat{MMD}^2$ is a U-statistic [Hoeffding, 1963, Peel et al., 2010] with "kernel" q:

$$q((x_i, y_i), (x_j, y_j)) = k(x_i, x_j) + k(y_i, y_j) - k(x_i, y_j) - k(x_j, y_i)$$

Kernel Test for the Two-Sample Problem

Observe samples $S_X = \{x_i\}_{i=1}^{n_x} \sim P$ and $S_Y = \{y_i\}_{i=1}^{n_y} \sim Q$

- $\mathbf{H_0} : \mathbf{P} = \mathbf{Q}$ (null hypothesis), $\mathbf{H_A} : \mathbf{P} \neq \mathbf{Q}$ (alternative hypothesis)
- Compute the value of the statistic $\widehat{MMD}(S_X, S_Y, \mathcal{F})$ and if
  - beyond some threshold θ: reject $\mathbf{H_0}$
  - otherwise: do not reject $\mathbf{H_0}$

Kernel embedding wrap-up and more

One thing to remember

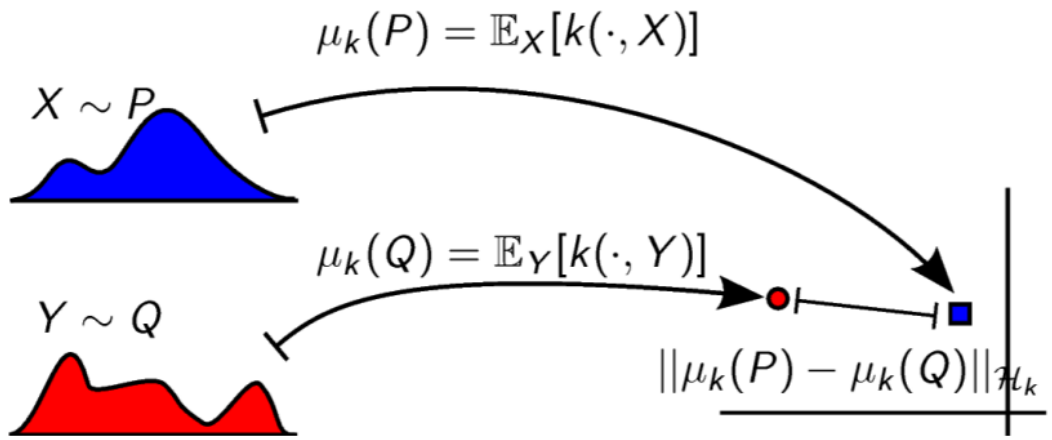


image from D. Sejdinovic, 2014

Extensions, uses

- Testing for independence (HSIC) [Gretton et al., 2005]
- Testing for conditional independence [Song et al., 2009]
- Testing for multivariate interaction [Sejdinovic et al., 2013]
- Kernel Bayes' rule [Fukumizu et al., 2013]
- Importance Sampling and sample bias correction [Huang et al., 2007]
- ...

Outline

Kernel Embedding of Distributions
- Example: the 2-sample test problem
- Kernel Embedding of Distributions

Large Scale Kernel Methods
- Random Features for Kernel Machines
- Fast optimization methods

Random Features for Kernel Machines

based on [Rahimi and Recht, 2007, Le et al., 2013]

Kernels methods for large-scale problems

Example: Image Net Challenge

- Number of training examples $m = 1,200,000$ in ILSVRC1000 dataset.
- Dimension d of encoded features for most algorithms is more than $20,000$
- Big data: d is large, m is huge

Features of Kernel Methods

	Linear kernel	Nonlinear kernel
Training speed	**very fast**	very slow
Training scalability	**very high**	low
Testing speed	**very fast**	very slow
Testing accuracy	lower	**higher**

How to get bold (using properties of the kernels considered)?

Naïve Kernel Methods

Base computational demands

- d: input feature dimension (e.g., $d = 20,000$)
- m: number of examples (e.g., $d = 1,200,000$)
- n: number of nonlinear basis function (e.g., $n = 120,000$)

Kernel expansion:

$$f(x) = \langle w, \Phi(x) \rangle = \left\langle \sum_{i=1}^{n} \alpha_i \Phi(x_i), \Phi(x) \right\rangle = \sum_{i=1}^{n} \alpha_i k(x_i, x)$$

number of features n is (often) linear with the number of examples m

	CPU Training	CPU Test	RAM Training	RAM Test
Naive	$O(m^2 d)$	$O(md)$	$O(md)$	$O(md)$

Kernel Methods with Random Features

- Random features & Linear in the feature space
  - $f(x) = \sum_{i=1}^{n} \alpha_i \hat{\Phi}(x, w_i)$
  - n random features – $n \ll m$
  - $\hat{\Phi}(x, w_i)$ is a feature function obtained via randomization

	CPU Training	CPU Test	RAM Training	RAM Test
Naive	$O(m^2 d)$	$O(md)$	$O(md)$	$O(md)$
Reduced set	$O(m^2 d)$	$O(nd)$	$O(md)$	$O(nd)$
Low rank	$O(mnd)$	$O(nd)$	$O(nd)$	$O(nd)$
RKS	$O(mnd)$	$O(nd)$	$O(nd)$	$O(nd)$
Fastfood	$O(mn \log d)$	$O(n \log d)$	$O(n)$	$O(n)$

Random Kitchen Sinks (RKS): Approximating Kernels

Bochner's Theorem

A kernel $k(x-y)$ on $\mathbb{R}^d$ is PSD if and only if $k(x-y)$ is the Fourier transform of a non-negative measure $p(w)$:

$$k(x-y) = \int_{\mathbb{R}^d} p(w) e^{j\langle w,(x-y)\rangle}\, dw$$

Consequence: approximating kernel computations

- For $w_1, \ldots, w_n$ IID according to $p(w)$

$$k(x-y) \approx \frac{1}{n}\sum_{i=1}^{n} e^{j\langle w_i,(x-y)\rangle} = \frac{1}{n}\sum_{i=1}^{n} e^{j\langle w_i,x\rangle} e^{-j\langle w_i,y\rangle} = \frac{1}{n}\sum_{i=1}^{n} z_{w_i}(x) z_{w_i}^*(y)$$

  where $z_w(x) = \exp(j\langle w, x\rangle)$.

- If $\phi(x) = [z_{w_1}(x) \cdots z_{w_n}(x)]^\top$ then

$$k(x-y) \approx \langle \phi(x), \phi(y)\rangle.$$

  - For Gaussian RBFs, we draw w from a Gaussian distribution
  - Computation: $O(nd)$ CPU & $O(nd)$ RAM

Random Kitchen Sinks (RKS) with Kernel Ridge Regression

```
% Training
W = randn(n, size(X,1));
phi= exp(i*W*x);
alpha = (phi*phi + eye(T)*noise)\(phi*y);

% Testing
ytest = alpha*exp(i*W*xtest);
```

from http://www.keysduplicated.com/~ali/random-features/

	CPU Training	CPU Test	RAM Training	RAM Test
RKS	$O(mnd)$	$O(nd)$	$O(nd)$	$O(nd)$

Massaging Wx

- Is there a way to speed up the computation of Wx?
- Is there a way to lower the memory needs to store W?

Answers: yes, Fastfood [Le et al., 2013]

Fastfood

Spoilers

- Wx can be computed in $O(n \log d)$ (instead of $O(nd)$)
- W need not be stored

	CPU Training	CPU Test	RAM Training	RAM Test
RKS	$O(mnd)$	$O(nd)$	$O(nd)$	$O(nd)$
Fastfood	$O(mn \log d)$	$O(n \log d)$	$O(n)$	$O(n)$

Fastfood

The magic recipe

Instead of W, use $\widetilde{W} = SHG\Pi HB$ where

- S is random diagonal scaling matrix (deals with spectrum)
- H is Walsh-Hadamard matrix admitting $O(d \log d)$ multiply
- G is random diagonal Gaussian matrix
- Π is random permutation matrix
- B is random binary $\{-1, 1\}$ diagonal matrix

so that $\widetilde{W}$ is approximately a Gaussian matrix

Crux: H can be built recursively (think of FFT)

$$H_{2n} = \begin{bmatrix} H_n & H_n \\ H_n & -H_n \end{bmatrix} \text{ and } H_1 = 1$$

Also

- $HG\Pi HB$ produces pseudo-random Gaussian vectors (of identical length)
- S fixes the lengths to have the correct distribution

Fastfood Results

Computing $\widetilde{W}x$

d	n	Fastfood	RKS	Speedup	RAM
1024	16384	0.00058s	0.0139s	24x	256x
4096	32768	0.00136s	0.1224s	90x	1024x
8196	65536	0.00268s	0.5360s	200x	2048x

Random features: partial conclusion and open questions

Conclusion

- It is possible to use features of the kernel at hand (cf. Bochner's theorem)
- Random features work well in practice
- Fastfood minimizes the storage and the computational demands

Open questions

- RKS & Fastfood transformations for other kinds of kernels?
- Things related to wavelet transforms?
- Improved generalization by implicit regularization?
- ...

Fast optimization methods

Optimization methods

- Chunking, decomposition methods
  [Osuna et al., 1997, Joachims, 1998, Platt, 1998]
- Bundle methods, cutting plane methods
  [Franc and Sonnenburg, 2009, Joachims et al., 2009, Teo et al., 2010]
- Active set strategies [Scheinberg, 2006]
- Stochastic gradient descent [Roux et al., 2012, Hardt et al., 2015]
- ...

Conceptual questions

- Tradeoffs of large-scale learning: optimization + approximation + estimation [Bottou and Bousquet, 2008]
- Distributed optimization and consequences on generalization
- Amortized-time integer programming
- Limited-bit representation
- ...

References I

Bottou, L. and Bousquet, O. (2008).

The tradeoffs of large scale learning.

In Platt, J., Koller, D., Singer, Y., and Roweis, S., editors, *Advances in Neural Information Processing Systems*, volume 20, pages 161–168. NIPS Foundation (http://books.nips.cc).

Fortet, R. and Mourier, E. (1953).

Convergence de la réparation empirique vers la réparation théorique.

Ann. Scient. École Normale Suprieure, 70:266–285.

Franc, V. and Sonnenburg, S. (2009).

Optimized cutting plane algorithm for large-scale risk minimization.

Journal of Machine Learning Research, 10:2157–2192.

Fukumizu, K., Song, L., and Gretton, A. (2013).

Kernel bayes' rule: Bayesian inference with positive definite kernels.

Journal of Machine Learning Research, 14:3753–3783.

Gretton, A., Borgwardt, K., Rasch, M., Schölkopf, B., and Smola, A. (2007).

A kernel method for the two-sample problem.

In *Adv in Neural Information Processing Systems 19*, pages 513–520. MIT Press.

References II

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schlkopf, B., and Smola, A. (2012).

A kernel two-sample test.

Journal of Machine Learning Research, 13:723–773.

Gretton, A., Bousquet, O., Smola, A., and Schölkopf, B. (2005).

Measuring statistical dependence with hilbert-schmidt norms.

In *Proc. of Algorithmic Learning Theory*, pages 63–77. Springer-Verlag.

Hardt, M., Recht, B., and Singer, Y. (2015).

Train faster, generalize better: Stability of stochastic gradient descent.

Hoeffding, W. (1963).

Probability inequalities for sums of bounded random variables.

Journal of the American Statistical Association, 58(301):13–30.

Huang, J., Smola, A., Gretton, A., Borgwardt, K., and Schölkopf, B. (2007).

Correcting sample selection bias by unlabeled data.

In *Adv. in Neural Information Processing Systems 19*.

References III

Joachims, T. (1998).

Making Large-Scale Support Vector Machine Learning Practical.

In Schölkopf, B., Burges, C., and Smola, A., editors, *Adv. in Kernel Methods – Support Vector Learning*, pages 169–184. MIT Press, Cambridge, MA.

Joachims, T., Finley, T., and Yu, C.-N. (2009).

Cutting-plane training of structural svms.

Machine Learning, 77(1):27–59.

Le, Q., Sarlos, T., and Smola, A. (2013).

Fastfood – approximating kernel expansions in loglinear time.

In *Int. Conf. in Machine Learning*.

Osuna, E., Freund, R., and Girosi, F. (1997).

Improved Training Algorithm for Support Vector Machines.

In *Proc. IEEE Workshop on Neural Networks for Signal Processing*, pages 276–285.

Peel, T., Anthoine, S., and Ralaivola, L. (2010).

Empirical bernstein inequalities for u-statistics.

In *Advances in Neural Information Processing Systems 23 (NIPS 2010)*.

References IV

Platt, J. (1998).

Sequential Minimal Optimization: a Fast Algorithm for Training Support Vector Machines.

Technical Report 98-14, Microsof Research.

Rahimi, A. and Recht, B. (2007).

Random features for large-scale kernel machines.

In *Adv. in Neural Information Processing Systems 19*.

Roux, N. L., Schmidt, M., and Bach, F. R. (2012).

A stochastic gradient method with an exponential convergence rate for finite training sets.

In *Advances in Neural Information Processing Systems 25*, pages 2663–2671. Curran Associates, Inc.

Scheinberg, K. (2006).

An efficient implementation of an active set method for svm.

J. of Machine Learning Research, 7:2237–2257.

Sejdinovic, D., Gretton, A., and Bergsma, W. (2013).

A kernel test for three-variable interactions.

In *Adv. in Neural Information Processing Systems 26*.

References V

Song, L., Huang, J., Smola, A., and Fukumizu, K. (2009).

Hilbert space embeddings of conditional distributions with applications to dynamical systems.

In *Proc. of the 26th Int. Conf. on Machine Learning*, pages 961–968.

Teo, C. H., Vishwanathan, S. V. N., Smola, A. J., and Le, Q. V. (2010).

Bundle methods for regularized risk minimization.

Journal of Machine Learning Research, 11:311–365.

Kernel Approximation

http://www.stat.ucdavis.edu/~chohsieh/teaching/ECS289G_Fall2016/lecture9.pdf

Kernel Approximation

- Kernel methods are hard to scale up because

  Kernel matrix G is an n-by-n matrix

- Can we approximate the kernel using a low-rank representation?
- Two main algorithms:
  - Nystrom approximation: Approximate the kernel matrix
  - Random Features: Approximate the kernel function

Kernel Matrix Approximation

- We want to form a low rank approximation of $G \in \mathbb{R}^{n \times n}$,
  where $G_{ij} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$

- Can we do SVD?

Kernel Matrix Approximation

- We want to form a low rank approximation of $G \in \mathbb{R}^{n \times n}$,

  where $G_{ij} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$

- Can we do SVD?

  No, SVD needs to form the n-by-n matrix

  $O(n^2)$ space and $O(n^3)$ time

Kernel Matrix Approximation

- We want to form a low rank approximation of $G \in \mathbb{R}^{n \times n}$,
  where $G_{ij} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$

- Can we do SVD?

  No, SVD needs to form the n-by-n matrix

  $O(n^2)$ space and $O(n^3)$ time

- Can we do matrix completion or matrix sketching?

Kernel Matrix Approximation

- We want to form a low rank approximation of $G \in \mathbb{R}^{n \times n}$,
  where $G_{ij} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$

- Can we do SVD?

  No, SVD needs to form the n-by-n matrix

  $O(n^2)$ space and $O(n^3)$ time

- Can we do matrix completion or matrix sketching?

  Main problem: need to generalize to new points

Nystrom Approximation

- Nystrom approximation:
  - Randomly sample m columns of G:

$$G = \begin{bmatrix} W & G_{12} \\ G_{21} & G_{22} \end{bmatrix}$$

W: m-by-m square matrix (observed)
G_{21}: $(n-m)$-by-m matrix (observed)
$G_{12} = G_{21}^T$ (observed)
G_{22}: $(n-m)$-by-$(n-m)$ matrix (not observed)

  - Form a kernel approximation based on these mn elements

$$G \approx \tilde{G} = \begin{bmatrix} W \\ G_{21} \end{bmatrix} W^{\dagger} \begin{bmatrix} W & G_{12} \end{bmatrix}$$

$W^{\dagger}$: pseudo-inverse of W

Nystrom Approximation

- Why $W^{\dagger}$?

  Exact recover the top left m-by-m matrix

- The kernel approximation:

$$\begin{bmatrix} W & G_{12} \\ G_{21} & G_{22} \end{bmatrix} \approx \begin{bmatrix} W \\ G_{21} \end{bmatrix} W^{\dagger} \begin{bmatrix} W & G_{12} \end{bmatrix} = \begin{bmatrix} W & G_{12} \\ G_{21} & G_{21} W^{\dagger} G_{12} \end{bmatrix}$$

Nystrom Approximation

Algorithm:

1. Sample m "landmark points": $\boldsymbol{v}_1, \cdots, \boldsymbol{v}_m$
2. Compute the kernel values between all training data to landmark points:

$$C = \begin{bmatrix} W \\ G_{21} \end{bmatrix} = \begin{bmatrix} K(\boldsymbol{x}_1, \boldsymbol{v}_1) & \cdots & K(\boldsymbol{x}_1, \boldsymbol{v}_m) \\ K(\boldsymbol{x}_2, \boldsymbol{v}_1) & \cdots & K(\boldsymbol{x}_2, \boldsymbol{v}_m) \\ \vdots & \vdots & \vdots \\ K(\boldsymbol{x}_n, \boldsymbol{v}_1) & \cdots & K(\boldsymbol{x}_n, \boldsymbol{v}_m) \end{bmatrix}$$

3. Form the kernel approximation $\tilde{G} = CW^{\dagger}C^T$
   (No need to explicitly form the n-by-n matrix)

- Time complexity: mn kernel evaluations ($O(mnd)$ time if use classical kernels)
- How to choose m?

  Trade-off between accuracy v.s computational time and memory space

Nystrom Approximation: Training

- Solve the dual problem using low-rank representation.
- Example: Kernel ridge regression

$$\min_{\boldsymbol{\alpha}} \boldsymbol{\alpha}^T \tilde{G} \boldsymbol{\alpha} + \lambda \|\boldsymbol{\alpha}\|^2 - \mathbf{y}^T \boldsymbol{\alpha}$$

  Solve a linear system $(\tilde{G} + \lambda I)\boldsymbol{\alpha} = \mathbf{y}$
- Use iterative methods (such as CG), with fast matrix-vector multiplication

$$(\tilde{G} + \lambda I)\boldsymbol{p} = C W^{\dagger} C^T \boldsymbol{p} + \lambda \boldsymbol{p}$$

  $O(nm)$ time complexity per iteration

Nystrom Approximation: Training

- Another approach: reduce to linear ERM problems!
- Rewrite as
$$\tilde{G} = CW^{\dagger}C^{T} = C(W^{\dagger})^{1/2}(W^{\dagger})^{1/2}C^{T}$$
- So the approximate kernel can be represent by linear kernel with feature matrix $C(W^{\dagger})^{1/2}$:
$$\tilde{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \tilde{G}_{ij} = \boldsymbol{u}_i^{T}\boldsymbol{u}_j,$$
where $\boldsymbol{u}_j = (W^{\dagger})^{1/2}\begin{bmatrix} K(\boldsymbol{x}_j, \boldsymbol{v}_1) \\ \vdots \\ K(\boldsymbol{x}_j, \boldsymbol{v}_m) \end{bmatrix}$
- The problem is equivalent to a linear models with features $\boldsymbol{u}_1, \cdots, \boldsymbol{u}_n \in \mathbb{R}^m$
  - Kernel SVM $\Rightarrow$ Linear SVM with m features
  - Kernel Ridge Regression $\Rightarrow$ Linear Ridge regression with m features

Nystrom Approximation: Prediction

- Given the dual solution $\boldsymbol{\alpha}$.
- Prediction for testing sample $\boldsymbol{x}$:

$$\sum_{i=1}^{n} \alpha_i \tilde{K}(\boldsymbol{x}_i, \boldsymbol{x}) \quad \text{or} \quad \sum_{i=1}^{n} \alpha_i K(\boldsymbol{x}_i, \boldsymbol{x}) \ ?$$

Nystrom Approximation: Prediction

- Given the dual solution $\boldsymbol{\alpha}$.
- Prediction for testing sample $\boldsymbol{x}$:

$$\sum_{i=1}^{n} \alpha_i \tilde{K}(\boldsymbol{x}_i, \boldsymbol{x})$$

- **Need to use approximate kernel instead of original kernel!**
  - Approximate kernel gives much better accuracy!
  - Because training & testing should use the same kernel.
- Generalization bound:
  - (Alaoui and Mahoney, "Fast Randomized Kernel Methods With Statistical Guarantees". 2014.)
  - (Rudi et al., "Less is More: Nystrom Computational Regularization". NIPS 2015.)
  - (using small number of landmark points can be viewed as a regularization)

Nystrom Approximation: Prediction

- How to define $\tilde{K}(\boldsymbol{x}, \boldsymbol{x}_i)$?

$$\tilde{G} = \begin{bmatrix} K(\boldsymbol{x}_1, \boldsymbol{v}_1) & \cdots & K(\boldsymbol{x}_1, \boldsymbol{v}_m) \\ \vdots & \vdots & \vdots \\ K(\boldsymbol{x}_n, \boldsymbol{v}_1) & \cdots & K(\boldsymbol{x}_1, \boldsymbol{v}_m) \\ K(\boldsymbol{x}, \boldsymbol{v}_1) & \cdots & K(\boldsymbol{x}_1, \boldsymbol{v}_m) \end{bmatrix} W^{\dagger} \begin{bmatrix} K(\boldsymbol{v}_1, \boldsymbol{x}_1) & \cdots & K(\boldsymbol{v}_1, \boldsymbol{x}) \\ \vdots & \vdots & \vdots \\ K(\boldsymbol{v}_m, \boldsymbol{x}_1) & \cdots & K(\boldsymbol{v}_m, \boldsymbol{x}) \end{bmatrix}$$

- Compute $\boldsymbol{u} = (W^{\dagger})^{1/2} \begin{bmatrix} K(\boldsymbol{v}_1, \boldsymbol{x}) \\ \vdots \\ K(\boldsymbol{v}_m, \boldsymbol{x}) \end{bmatrix}$ (need m kernel evaluations)
- Approximate kernel can be defined by

$$\tilde{K}(\boldsymbol{x}, \boldsymbol{x}_i) = \boldsymbol{u}^T \boldsymbol{u}_i$$

Nystrom Approximation: Prediction

- Prediction:

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i \boldsymbol{u}^T \boldsymbol{u}_i = \boldsymbol{u}^T (\sum_{i=1}^{n} \alpha_i \boldsymbol{u}_i)$$

  $(\sum_{i=1}^{n} \alpha_i \boldsymbol{u}_i)$ can be pre-computed
  $\Rightarrow$ prediction time is m kernel evaluations

- Original kernel method: need *n* kernel evaluations for prediction.
- Summary:

	Quality	Training time	Prediction time
Original kernel	exact	$n^{1.x}$ kernel + kernel training	n kernel
Nystrom (m)	rank m	nm kernel + linear training	m kernel

Nystrom Approximation

- How to select landmark points $(\boldsymbol{v}_1, \cdots, \boldsymbol{v}_m)$
  - Traditional approach: uniform random sampling from training data
  - Importance sampling (leverage score):
    - Drineas and Mahoney "On the Nystrom method for approximating a Gram matrix for improved kernel-based learning". JMLR 2015.
    - Gittens and Mahoney "Revisiting the Nystrom method for improved large-scale machine learning". ICML 2013
  - Kmeans sampling (performs extremely well) :
    Run kmeans clustering and set $\boldsymbol{v}_1, \cdots, \boldsymbol{v}_m$ to be cluster centers
    - Zhang et al., "Improved Nystrom low rank approximation and error analysis". ICML 2008.
  - Subspace distance:
    - Lim et al., "Double Nystrom method: An efficient and accurate Nystrom scheme for large-scale data sets". ICML 2015.

Nystrom Approximation (other related papers)

- Distributed setting: (Kumar et al., "Ensemble Nystrom method". NIPS 2009).
- Block diagonal + low-rank approximation: (Si et al., "Memory efficient kernel approximation". ICML 2014.)
- Nystrom method for fast prediction: (Hsieh et al., "Fast prediction for large-scale kernel machines. " NIPS, 2014.)
- Structured landmark points: (Si et al., "Computational Efficient Nystrom Approximation using Fast Transforms". ICML 2016.)